Express Mail Label # EK774863271US

Patent Application of

Kendyl A. Román

for

TITLE: FASTER IMAGE PROCESSING

BACKGROUND–FIELD OF THE INVENTION

This invention relates to image processing.

BACKGROUND–RELATED TECHNOLOGY

ANSI Standard C "memcpy" Function

A given computer hardware architecture will have an optimal means of copying a block of data from one location in a memory to another location. Complex Instruction Set Computing (CISC) architectures implement instructions that over a number of CPU cycles move a block of data. Reduced Instruction Set Computing (RISC) architectures optimize the instruction set to process each instruction in one or two CPU cycles but also included instructions that can be used to implement a short routine that will accomplish the block move in an optimal manner. An efficient routine for copying a block of data can be implemented for each specific computer architecture.

Some computer architectures include Direct Memory Access (DMA) circuitry that transfers data between memory and input/output (I/O) devices without continual central processing unit (CPU) intervention.

The ANSI standard for the C Programming Language defines a "memcpy" library function as an interface to an efficient routine for copying a block of bytes to another location.

Graphical Images

A television screen has a 4:3 aspect ratio. In the United States, television signals contain 525 scan lines of which 480 lines are visible on most televisions. When an analog video signal is digitized, each of the 480 lines are sampled 640 times, and each sample is represented by a number. Each sample point is called a picture element, or pixel. A two dimensional array is created that is 640 pixels wide and 480 pixels high. This 640 x 480 pixel array is a still graphical image that is considered to be full frame. The human eye can optimally perceive approximately 16.7 thousand colors. A pixel value comprised of 24 bits can represent each perceivable color. A graphical image made up of 24-bit pixels is considered to be full color. A standard Super VGA (SVGA) computer display has a screen resolution of 640 by 480 pixel. Twenty-four bits is three bytes. It is common to use a fourth byte for each pixel to specify a mask value or alpha channel. A typical image being processed may contain over 1.2 million bytes of data.

When digitizing a video signal, or when manipulating the graphics to be output as a video signal or to be displayed on a computer display it may be necessary to copy the image data to another area of memory (a buffer) for some type of image processing. However, the copied buffer takes up significant memory resources. Also the time it takes to copy the image can be significant especially when the image processing must be done in real time. Those skilled in the art realize that to improve processing performance the number of

memory buffers containing a copy of the same data should be reduced to the minimum set possible.

Display Video RAM

The memory of a computer system may be physically implemented in different areas or on different boards. The main memory is used for storage of program instructions and data. A special memory area called "video RAM" may be dedicated to storing the image that is to be displayed on the computer display. The video RAM has special hardware that allows it to be accessed to update the display over 60 times a second.

Capture Video RAM

A video digitizer or video capture card may also contain a special memory area similar to display video RAM for capturing the digital samples from the video signal. This RAM may also have special hardware that allows it to be updated 60 times a second.

Cache Memory

Many computer architectures implement one or more levels of memory caching whereby blocks of memory data are stored in a cache memory that may be accessed more rapidly by the CPU. Typically input and output (I/O) memories such as video RAM, capture RAM, or hard disk buffers are not cached.

SUMMARY OF THE INVENTION

In accordance with the present invention, methods are provided of increasing performance of image processing by copying image data between I/O memory and main memory where CPU intensive processing of the image data is more efficiently performed.

Objects and Advantages

Accordingly, beside the objects and advantages of the method described in the patent above, some additional objects and advantages of the present invention are:

(a) to provide efficient processing of image data prior to display on a computer display.

(b) to provide efficient processing of image data being captured in real time with a video digitizer.

(c) to reduce the time necessary to process the image data.

DRAWING FIGURES

In the drawings, closely related figures have the same number but different alphabetic suffixes.

Fig 1A shows a basic computer architecture.

Fig 1B shows components for video digitizing.

Fig 1C shows components for computer display.

Fig 2A shows a multi-level cache architecture.

Fig 2B shows a DMA architecture.

Fig 3A shows images copied between an I/O video RAM and main memory.

Fig 3B shows an input image being input and encoded.

Fig 3C shows encoded data being decoded and output.

Fig 4 shows row by row copy of a subset image.

Fig 5 shows a flowchart for copying a subset image from I/O RAM to a memory buffer.

Fig 6 shows a flowchart for copying a memory buffer to a subset image in I/O RAM.

Reference Numerals in Drawings

100	input	101	CPU
102	output	103	memory
110	video source	111	video digitizer
113	capture video RAM	120	display video RAM
121	video display		

220	I/O RAM	230	cache
240	CPU cache	250	DMA circuitry
252	DMA control	254	DMA-Memory bus
256	DMA-I/0 bus		
300	buffer	305	buffer-image copy
310	image	320	encoder
330	encoded data	340	decoder
400	super-image	420	first image line
422	second image line	424	last image line
430	first buffer line	432	second buffer line
434	last buffer line		
w	image width	h	image height
x	image horizontal offset	y	image vertical offset
500	image copy start	510	image copy initialization step
520	set counter step	530	image copy done decision
540	image copy step	550	update pointers step
560	increment index step	599	image copy exit
600	buffer copy start	610	buffer copy initialization step
620	set counter step	630	buffer copy done decision
640	buffer copy step	650	update pointers step
660	increment index step	699	buffer copy exit

DESCRIPTION OF THE INVENTION

Fig 1A to 1C–Computer Architectures

Fig 1A is a block diagram showing the basic components of a computer, comprising an input 100, a CPU 101, an output 102, and memory 103.

Fig 1B shows an embodiment of the computer input 100 specialized to input video data. A video source 110 is connected to a video digitizer 111. The video digitizer 111 converts the analog video signal from the video source 110 to a digital format. Some video digitizers transfer the video data to memory 103 for storage. Alternatively, some video digitizers contain capture video RAM 113 which can store the captured video data on the video digitizer 111 hardware without using memory 103 for storage.

Fig 1C shows an embodiment of the computer output 102 specialized to output video data. A video display 121 (also knows as a computer monitor) displays graphical information based on data contained in a display video RAM 120. Programs running on the CPU 101 determine the contents of the display video RAM that is then shown by pixels on the video display 121.

Fig 2A and 2B–Caching and DMA

Fig 2A is a block diagram showing computer architecture where there optionally are two levels of caches. The CPU 101 has an internal cache known as a CPU cache 240 that can store copies of recently accessed memory blocks that contain program instructions or data. A cache 230 stores copies of recently accessed blocks of memory data, because the cache 230 is outside the processor it is sometimes referred to as an external cache. In an architecture where the CPU has an internal CPU cache 240, the cache 230 may also be referred to as a level 2 cache.

If a copy of a block of memory data is in the CPU cache 240 or the memory cache 230, the CPU 101 can access it much faster than if the data has to be fetched from memory 103. If the data is not available to the CPU 101, the CPU 101 stalls causing there to be cycles where no useful processing is being done. The use of caches (230, 240) can have a significant impact of the speed of data processing.

It is common for input and output device registers and memories to be mapped into the memory address range. This is called memory mapped I/O. In a computer architecture

that uses memory mapped I/O, the random access memory (RAM) associated with computer input 100 and output 102 devices can be accessed by programs running on the CPU as if they were memory 103 RAM. Because the I/O RAM 220 can be modified by its respective input 100 or output 102 device, special provisions are made so that the blocks of memory from I/O RAM 220 are not stored in the cache 230 or the CPU cache 240 (or if they are stored in the cache they are marked as invalid so that the CPU will fetch the current contents of the I/O RAM 220 rather than use the obsolete data in the cache). Examples of I/O RAM 220 include capture video RAM 113 and display video RAM 120.

Fig 2B shows a computer architecture with direct memory access (DMA) circuitry 250. Without DMA circuitry 250, the CPU 101 must be involved in transferring data from memory 103 to I/O RAM 220. This CPU involvement takes the CPU 101 processing power away from executing other program instructions and adds overhead to handle the interruptions. DMA circuitry 250 is used to copy blocks of data directly between memory 103 and I/O RAM 220. A DMA operation is initiated by the CPU 101 with a DMA control 252 sent from the CPU 103 to the DMA circuitry 250. Once the DMA operation is initiated the CPU can return to other work. The DMA circuitry moves the data from memory 103 to I/O RAM 220 along the DMA-memory bus 254 and DMA-I/O bus 256 or from I/O RAM 220 to memory 103. In practice, the DMA circuitry may become a secondary bus master of a system bus that interconnects the CPU 101, I/O RAM 220, and memory 103. Once the data transfer is complete the DMA circuitry 250 notifies the CPU.

Processing Speed Improvement–Fig 3A to 3C

When video data is being displayed or captured the storage (memory 103 or I/O RAM 220) holding the data is continually being accessed by the video display circuitry or video digitizing circuitry. Also the capture video RAM 113 and the display video RAM 120 typically is not cached by a CPU 101 in any cache (230 or 240), so when processing the

video data for compression, encryption, enhancement, or decompression it is significantly faster to process the data in cacheable main memory.

The present invention uses a memory copy function (similar to a memcpy function or a substantially similar set of computer instructions) to copy the desired image data from an I/O RAM 220 to a cacheable main memory 103 (Fig 2A) where it can be more efficiently processed. After the processing is done, the processed image is then copied back to the display video RAM 120 for display on the video display 121 (Fig 1C).

Fig 3A shows a buffer 300 in memory 103 and an image 310 stored in I/O RAM 220. The buffer-image copy 305 of data between the buffer 300 and the image 310 is shown as bi-directional arrows. Once the image data is copied from the image 310 to the memory buffer 300 it can be much more efficiently processed by the CPU 103. Fig 3B shows an encoder 320 program which accesses the buffer 300 applying enhancement, compression, or encryption algorithms as needed to produce encoded data 330. The encoded data 330 can be stored on a storage device or transferred over a network to another computer. Fig 3C shows a decoder 340 program processing the encoded data 330 into another instance of a memory buffer 300. The decoder can decrypt, decompress, or enhance the encoded data as needed and place the resulting data in a memory buffer 300.

This invention discovered that is was much more efficient to write the decoded data to a memory buffer 300 instead of writing it directly to image 310 in I/O RAM 220 as each pixel is processed. Once the decoder processing is complete, the buffer-image copy 305 is used to transfer the data from the buffer 300 to the I/O RAM 220. The I/O RAM could be a display video RAM 120 as shown in Fig 1C.

Not Obvious

The speed improvement yielded by this invention was not obvious to one skilled in the art of computer programming. The video data is large, up to 1.2 million bytes, and the time to copy it from one buffer to another generally is thought to be overhead that will

decrease performance. This invention teaches that because of hardware lockout, collisions with the video circuitry, the lack of data caching in the CPU cache 240 or memory cache 230, or other factors, the extra copy can significantly reduce the processing time, and thus reduce the overall time required to process the data and to display or capture the video data.

The memory copy routine used in the buffer-image copy 305 may use processor specific code, or other methods, to move blocks of data between the memory 103 (or the caches (230, 240)) and the I/O RAM 220.

The methods of this invention are much more efficient (due to I/O RAM lockouts and conflicts) than processing each pixel a byte or word at a time in place in I/O RAM 220.

Alternatively, DMA circuitry 250 (Fig 2B) may be used to increase the speed of transfer between memory 103 and the I/O RAM 220.

In one embodiment of this invention the entire image is copied by a single call to the memcpy function. This has the advantage of only making one function call.

Fig 4–Preferred Embodiment

In the preferred embodiment, only a subset image 310 of the data in I/O RAM 220 is of interest for processing, so the memory copy function is called repeatedly to copy each line of desired image data. For example if the desired subset is 320 by 240, the memory copy function is called 240 times and copies 320 pixels each time. This has the advantage of only copying the desired data. Even though there is more overhead in determining how to copy the subset and in calling the memory copy function multiple time, the time saved by copying less data more than compensates for the additional overhead. Less memory is used to hold the main memory buffer and less data must be processed.

Fig 4 is a diagram of the buffer 300 and the image 310 that shows more detail than Fig 3A. The subset image 310 is contained within a super-image 400. When a television video signal is digitized there are portions of the signal that are not visible on most television displays. The video digitizer often will process all of the video signal producing a

super-image 400 that contains data that surrounds the subset image 310 and the surround data typically is of no interest. If the origin of the super-image 400 is (0, 0) the image 310 of interest can be found at a coordinate (x, y) composed of the image horizontal offset x and the image vertical offset y. The image width w and the image height can be used to allocate the memory buffer 300, rather than copying the entire super-image 400. The coordinate of the last pixel of the desired image is (x+w, y+h).

In the preferred embodiment, the first image line 420 (starting at (x,y)) is copied (305) to the first buffer line 430 for the length of the image width w. Next the second image line 422 is copied to the second buffer line 432. Each line is copied until the last image line 424 is copied to the last buffer line 434. After the desired data is copied in this manner the buffer 300 can be efficiently processed. Buffer 300 is smaller than the super-image 400 and the data of interest is contiguous so it can be processed more efficiently. Buffer 300 can be cached and will have typically no conflict from other accesses.

Fig 4 also illustrates the reverse process of copying a buffer 300 containing processed data to a super image 400 in an I/O RAM 220 (Fig 3A). Each line of the buffer 300 is copied (305) to the image 310 in the super image 400 at the desired offset (x,y). In this reverse process the first buffer line 430 is copied to the first image line 420. The second buffer line 432 is copied to the second image line 420, and so forth, until the last buffer line 434 is copied to the last image line 424. The same advantages of buffer 300 being smaller, contiguous, cacheable, and conflict free also apply to the reverse process.

Fig 5–Image Copy Flowchart

Fig 5 is a flow chart for the method of copying the image 310 to buffer 300 as shown in Fig 4. The method starts at an image copy start 500 entry point. Next an image copy initialization step 510 comprising the following is executed:

- the line size is set to the image width w.
- the number of lines is set to the image height h.

- the row size is calculated by dividing the total bytes in a row of the super image by the number of bytes per pixel.
- the copy size is calculated by multiplying the line size by the number of bytes per pixel.
- the source pointer is set the base address of the image 400 plus the calculation of the number of bytes to get to the (x,y) offset: ((y * row size + x) * bytes per pixel).
- the destination pointer is set to the base address of the buffer 300.

Next, in a set counter step 520, the row index is set to 0. An image copy done decision 530 is made by comparing the row index to the number of lines. If one or more lines still need to be copied, flow continues to an image copy step 540. In the image copy step 540, the memory copy function is called to copy copy-size bytes from the current source pointer to the current destination pointer (effectively copying a line of the image 310 to the buffer 300). Next, in an update pointers step 550, the source pointer is incremented by the number of bytes in a row of the super image (effectively addressing the beginning of the next line of the image 310), and the destination pointer is incremented by the number of bytes in a line of the buffer 300 (effectively addressing the beginning of the next line of the buffer 300). Next in an increment index step 560, the row index is increment. Flow continues to the image copy done decision 530, and the loop continues until each line of the image 310 is copied. When the image has been fully copied, flow terminates at an image copy exit 599 point.

Fig 6–Buffer Copy Flowchart

Fig 6 is a flow chart for the method of copying the buffer 300 to the image 310 as shown in Fig 4. The method starts at a buffer copy start 600 entry point. Next a buffer copy initialization step 610 comprising the following is executed:

- the line size is set to the image width w.

- the number of lines is set to the image height h.
- the row size is calculated by dividing the total bytes in a row of the super image by the number of bytes per pixel.
- the copy size is calculated by multiplying the line size by the number of bytes per pixel.
- the destination pointer is set the base address of the image 400 plus the calculation of the number of bytes to get to the (x,y) offset: ((y * row size + x) * bytes per pixel).
- the source pointer is set to the base address of the buffer 300.

Next, in a set counter step 620, the row index is set to 0. A buffer copy done decision 630 is made by comparing the row index to the number of lines. If one or more lines still need to be copied, flow continues to a buffer copy step 640. In the buffer copy step 640, the memory copy function is called to copy copy-size bytes from the current source pointer to the current destination pointer (effectively copying a line of the buffer 300 to the image 310). Next in an update pointers step 650, the destination pointer is incremented by the number of bytes in a row of the super image (effectively addressing the beginning of the next line of the image 310), and the source pointer is incremented by the number of bytes in a line of the buffer 300 (effectively addressing the beginning of the next line of the buffer 300). Next in an increment index step 660, the row index is increment. Flow continues to the buffer copy done decision 630, and the loop continues until each line of the buffer 300 is copied. When the buffer has been fully copied, flow terminates at a buffer copy exit 699 point.

Advantages

Execution Speed

The methods of the present invention provide a decrease in the processing time required to process images that are being input or output. This decrease in processing time

allows for video images to be enhanced, compressed, and encrypted in real time. The time saved by these methods can be used to execute more efficient compression algorithms that may in turn reduce the bandwidth required to transfer the encoded data between computers or may reduce the space needed to store the encoded data.

Reduced Memory Requirements

The selection of a subset image 310 from a super image 400 (Fig 4) reduces the amount of memory needed to hold the data being processed.

Conclusion, Ramification, and Scope

Accordingly, the reader will see that the methods the present invention provides a means of reducing the processing time and computer resources needed to process images being input or output.

Furthermore, the present invention has additional advantages in that it provides a means for reducing the space required in a storage medium.

Although the descriptions above contain many specifics, these should not be construed as limiting the scope of the invention but as merely providing illustrations of some of the preferred embodiments of this invention. For example, the memory copy algorithm can be implemented in a number of ways without limiting the scope of this invention to the use of a particular implementation.

Thus the scope of the invention should be determined by the appended claims and their legal equivalents, and not solely by the examples given.